ISO TC184/SC4/WG12 N512

First Edition: 2000-2-10

[File: Pc/Word]

ISO 10303 White Paper

Parametric Representation and Exchange: Comments on N440 : Parametrization and constraints for explicit geometric product models

ABSTRACT:

The series of International Standards to be produced by the project ISO 10303 WG12 / Parametrics deal with the representation and exchange of information associated with parametrization and constraints as applied to product model shape. It defines methodologies for vendor and implementation independent representation and exchange of parametric models.

The present document is a White Paper which summarizes comments and recommendations to N440: Parametrization and constraints for explicit geometric product model(Dec 1st 1999 version).

KEYWORDS: range specification, equi-value dimension group, fixed dimension, fixed entity, rigid set constraint, imported entity

COMMENTS TO READER:

This document is intended as a discussion document, and informed comment is invited, either via the SC4

Parametrics exploder or directly to the editor.

Project Leader: Michael J. Pratt
Address:
NIST
Building 220, Room A127
Gaithersburg, MD 20899-0001
USA
Telephone: +1 (301) 975-3951
Telefax: +1 (301) 975-4482
Electronic mail: pratt@nist.gov

Document Editor: Akihiko Ohtaka
Address:
Nihon Unisys, Ltd.
1-1-1 Toyosu, Koto-ku
Tokyo 135-8560 Japan
Telephone:+81 3 5546 4784
Telefax: +81 3 5546 7810
Electronic mail: akihiko.ohtaka@unisys.co.jp

Comments on WG12/N440: Parametrization and constraints for explicit geometric product models

First ed.: 10th Feb. 2000
Nihon Unisys, Ltd.
Akihiko Ohtaka

Contents

page

1. Introduction

This document summarizes comments and recommendations on WG12/N440: Parametrization and constraints for explicit geometric product models(Dec. 1st 1999 version).

2. Technical comments and recommendations

(1) **Issue**: Insufficient range specification

Specification of allowable ranges for model parameters is said to be within the scope of parametric expression schema in 1.1(page-2). But, explicit declaration of allowable ranges seems to be nowhere defined.

Recommendation: Relationships between parameters naturally limit a valid range of a parameter which appears in relationships, but it is not enough. Explicit declaration of allowable range for each parameter is necessary. Necessary variations for specifying allowable range of variable dimension used in most commercial systems are as follows.

1) Interval

Minimum and maximum values could be specified with numerical value input or by expression.

2) Rounding

When rounding value 10 is specified, 10, 20, 30, --- are allowable values.

3) Pitch

Specify starting value and incremental pitch

4) Set

Allowable values are described as a set

(2) **Issue**: Lack of global rule in parametric expression schema

Though individual types and entities are explained in some detail, basic rule or understanding for a set of expressions which is concerned with one product shape is not given. Major point we should consider is how we treat order of expressions. As already mentioned in my past document, there are two approaches in major commercial systems for this issue. Some system makes much of the order and solves given expressions with given order. In this approach, one variable may appear on the left hand side twice or more times and the last appearance becomes effective. The other approach is to solve a set of expressions irrespective of given order. If one variable which appears on the right hand side of some expression appears on the left hand side in the later expression, left hand side one is first solved, and then the result is assigned to the original expression. In order to get rid of ambiguity, one variable had better appear on the left side hand only once in the latter approach. Point here is not the solution method, but which is acceptable for users. Most users opinions are that they can explicitly declare relations of variables, but it is hard for them to be conscious of order of

expressions.

Recommendation: Statement on which approach this standard recommends had better be written. If we force one variable to appear only once on the left hand side or not needs further discussion.

(3) **Issue**: Use of assignment statement or not for equality expression

Different from the specification in this standard, some major CAD systems, at least Pro/E, I-DEAS and CADCEUS, use assignment statement for representing relation among dimension variables such as;

D1 := D2 + 3*D3 + F(u,v)

In this case, D2 and D3 are independent variables, but D1 is dependent on them. Therefore, value change of D1 is **not allowed**. This is typical 'directed constraint' in our terminology.

Discussion of (2) above reflects this fact. If we also adopt assignment statement in this standard or not should be discussed.

Recommendation: We need further discussion, but the author personally thinks that assignment statement is better for representing dependence relation among dimensions since design intent is more obviously represented. Combined use of assignment statement with equi-value dimension group defined later could represent all the design requirements we encountered.

(4) **Issue**: It is necessary to introduce equi-value dimension group

There are many cases that two or more dimensions always should take the same value though they have different appearances. Typical example is the case that radii of fillets scattered on a die surface should always take the same value from manufacturing requirement point of view.

If any one dimension within a group is changed, values of all the other dimensions in the same group are automatically changed the same. There is no dependence relation among dimensions within a group. Therefore, treatment of equi-value dimension group is more easier for a designer than assignment expression.

Recommendation: Introduce it. Possible EXPRESS specification follows.

EXPRESS specification:

```
*)
ENTITY equi-value_dimension_group
  name: label;
  current_value: express_expression;
  members: SET[2:?] OF variable_semantics;
  description: text;
END_ENTITY;
(*
```

(5) **Issue**: How to incorporate user defined function

There is no specification on how to incorporate user defined function in parametric_expression schema, which is a fundamental requirement for supporting relation among variables.

Recommendation: We need further discussion and should develop a mechanism to incorporate user defined functions.

(6) **Issue**: Cumbersome type definitions in parametric_expression schema.

There are 22 type definitions in the schema, which is too much for realizing simple requirements of representing variable ,expression and function for use in relating dimension variables.

Recommendation: Some case study is recommended to confirm if all of those types are Necessary or not.

(7) **Issue**: The situation of 'free form constraint' had better be reconsidered.

Explicit geometric constraint and free form constraint are constraints of the same abstraction level from data modeling point of view. I agree to define explicit constraint to summarize these in one level higher abstraction level, and to prepare future enhancement.

But, current situation of free form constraint within the explicit constraint does not satisfy above described structuring.

Recommendation: Free form constraint had better be separated to another schema.

(8) **Issue**: The difference of directed and undirected constraints

I basically agree with your statement and examples. But, there is one point which should be made clear. That is how we situate when some constraint parameter is changed, say, offset distance. Should we regard it as reference element change or constrained element change?

Recommendation: I personally think it is not reference element change nor constrained element change. It is a condition change. There should be some explanation on condition change.

Another recommendation is that we had better introduce clear notation difference of directed and undirected constraint in order to make readers understanding clear.

(9) **Issue**: Is 5.3: explicit_constraint type definitions necessary?

Typical use of variables and expressions&functions are as follows.

D1 := #21.nominal_value

D2 := #22.nominal_value

D3 := #45.radius

D1 := D2 + 3*D3 + F(u1,u2)

Function F could be built in function or user defined function. I am not sure if entity_instance_expression, explicit_constraint_expression, and dimension_relation_expression are necessary for realizing above requirement.

Recommendation: Some case study is appropriate for judging their necessity.

(10)**Issue**: Question on the attribute type of **explicit_constraint** entity

Explicit_constraint – **predefined_constraint** – **explicit_geometric_constraint** is entity hierarchy and constrained_elements and reference_elements defined as attributes of explicit_constraint is inherited to explicit_geometric_constraint. Question is if value_assignable_expression currently specified as the type of reference_elements and constrained_elements is appropriate or not. It is an express_expression, and therefore, an express_in_string, but express_in_string has too much macroscopic functionality.

In case we apply explicit geometric constraint, its attributes are limited types of geometric entities, and current definition seems to be too much macroscopic.

Recommendation: As mentioned before, we had better separate free form constraint. If it is done, there remains no need to inherit attributes from explicit_constraint to explicit_geometric_constraint. Therefore, we can define attribute types of explicit_geometric_constraint more naturally.

(11)**Issue**: Situation change of Clause-6:Schema explicit_geometric_constraint is required

Since Parametric assembly constraints(in other words, explicit_3D_geometric_constraints) specification is now provided in Clause-7 and Clause-8, the situation of Caluse-6 should be reconsidered. At present, Clause-6 describes both on 2D and 3D explicit geometric constraints, and only gives detailed specification of 2D constraints.

Recommendation: There are two possible resolutions. One is to merge Clause-7:Parametric assembly constraints into Clause-6. The other is separate out 3D related descriptions into Clause-7. Pros and cons are as follows.

<Merger case>

Pro-1: Paralell_constraint, for example, appears only once in this standard. Therefore, from removal of redundancy point of view, this approach is better.

Con-1: Main user of 2D constraint is a part designer, and that of 3D constraint is an assembly designer. The user is forced to extract only necessary portion from this standard for his job which is cumbersome.

Con-2: All the major parametric CAD systems separate 2D and 3D. Modules and users manuals are separated for making user's understanding easier.

<Separate case>

Totally opposite to merger case. Pros there is cons here and cons there is pros here.

The author recommends the latter, namely separate 2D and 3D. If we agree to my opinion, all the 3D related descriptions should be removed out from Clause-6. In that case, it may be better to change current schema name to '**explicit_2D_geometric_constraint**'. Recommended subtypes of explicit_constraint are then as follows.

explicit_constraint explicit_2D_geometric_constraint

explicit_3D_geometric_constraint(or parametric_assembly_constraint)

free_form_constraint

Comments which follow is based on this idea.

(12)**Issue**: Consideration-1 on 6.2.3: Dimensional constraints

Cited example discusses on the difference of dimensional constraint and logical constrain, but this example reminds me another view. Consider the following two cases.

1) #21.radius := #22.radius
2) #21.radius and #22.radius are members of the same **equi_value_dimension_group**

In 1) case, #21.radius is dependent on #22.radius. But, in 2) case, there is no dependence and they are always forced to have the same value.

Recommendation: None in particular. Just discussion to make all dimension related matters clear.

(13)**Issue**: Consideration-2 on fixed dimension related to 6.2.3: Dimensional constraints

Any dimension with a value in a geometrically constrained model is obviously constrained to have the value. Constraint solver is never allowed to change dimension value unless the use explicitly shows his intention to change the value within limited range. But, there are dimensions without allowable range. It means that the user want to fix those dimensions with current values. In order to satisfy this very natural design requirement, we need some entity to declare that specified dimension is fixed.

Recommendation:

There are two possible ways of satisfying this requirement. One is to provided fixed specification in the allowable range specification. The other is to introduce a new entity for that purpose. I prefer the latter since it is more explicit and suitable for the purpose of this standard. Proposed entity specification follows.

EXPRESS specification:

```
*)
ENTITY fixed_dimension_constraint
  members: SET[1:?] OF variable_semantics;
  description: text;
SUBTYPE OF (explicit_2D_geometric_constraint);
END_ENTITY;
(*
```

(14)**Issue**: Revision of 6.3: explicit_geometric_constraint type definitions

All the selective geometry types necessary for the parametric_assembly_constraint (or, explicit_3D_geometric_constraint) is defined in Clause-7. Therefore, geometry types which is defined here for use in 3D geometric constraint may be better to be removed unless 3D application other than parametric assembly constraint is intended..

Recommendation: Since geometry types necessary in sketch are limited to point and some type of curve, this section had better be revised accordingly. Probably, only the following geometry select type is required.

EXPRESS specification:

```
*)
TYPE point_or_curve_element = SELECT
  (point,
  curve);
END_TYPE;
(*
```

(15)**Issue**: Revision of 6.4.1: **explicit_geometric_constraint** entity is necessary

If we agree to my opinion to separate 2D and 3D geometric constraints, then this entity is a supertype entity of 2D geometric constraints. Therefore, owner attribute is not optional, but necessary. **Fixed_entity_constraint**(replacement of **fixed_point_constraint**) and **rigid_set_constraint**(new entity) as presented below are obviously subtypes of **explicit_geometic_constraint**. Therefore, they should be added in subtype description of this entity.

Recommendation: Revision as mentioned above is required.

(16)**Issue**: Target of fixed entity should be extended to include curves

All parametric CAD systems allow fix of any type of one or more entity which exists in a sketch. There is no reason to limit fixed constraint to points. Though a point is a defining entity of a curve, the user is forced to fix all points on a curve with current specification when he wants to fix the curve, which is not realistic.

Recommendation: Replace current fixed_point_constraint entity by the following entity.

EXPRESS specification:

```
*)
ENTITY fixed_entity_constraint
   SUBTYPE OF (explicit_geometric_constraint);
   SELF\explicit_constraint.constrained_elements : SET[1:?] OF point_or_curve_element;
END_ENTITY;
(*
```

(17)**Issue**: It is necessary to introduce rigid set constraint

There are cases that the user wants to keep a group of elements within a sketch unchanged though

the location and orientation of the group itself could be changed depending on constraints with elements outside of the group. It means that inside of the group is firmly fixed but the group as a whole behave as a rigid body. In order to satisfy this requirement, there should be a capability to group two or more sketch elements, and assign them rigid attribute.

Recommendation: The author recommends to introduce the following entity.

EXPRESS specification:

```
*)
ENTITY rigid_set_constraint
  SUBTYPE OF (explicit_geometric_constraint);
  name : label;
  SELF\explicit_constraint.constrained_elements : SET[2:?] OF point_or_curve_element;
END_ENTITY;
(*
```

(18)**Issue**: Treatment of direction in 6.4.21: parallel_offset_constraint_with_dimension questionable

Though direction_constraint appears as an optional attribute in entity description, direction itself is not explicitly given and it is said that it may be determined from the current result shape model. Usually, current result shape model is a complicated 3D model based on sketches, and it is hard to judge the direction from the current result.

Recommendation: Direction had better be given explicitly in this entity.

(19)**Issue**: Additional dependency constraints seem necessary.

As we already discusses, current Part42 includes procedural entity(in other words, entity dependent on other entity) which had better be shifted into parametrics specification if we want more clear representation of dependence. These are offset curve/surface, projected curve, intersection point/curve, and swept surface/solid.

Currently, only offset curve/surface, and projected curve are included in this specification.

Recommendation: From the balance of specification point of view, we had better add intersection point/curve and swept surface/solid.

(20)**Issue**: Insufficient implicit_curve functionality

As for essential functionality of imported point/curve, my document WG12/N408 clearly identifies functionality common in major parametric CAD systems. They are;

<when imported element on a sketch is a curve>

1) It was created by projecting an outside curve onto a sketch plane
2) It was created by projecting an outline edges of a surface or a solid model onto a sketch plane
3) It was created by calculating intersection of a surface model with a sketch plane

<when imported element on a sketch is a point>

1) It was created by projecting an outside point onto a sketch plane

2) It was created by calculation intersection of a curve with a sketch plane

6.4.33:implicit_curve does not fully satisfy above functionality.

Recommendation: Satisfaction of all requirements above is necessary.

(21)**Issue**: It is described in 6.4.24:Sketch that the entity definition of imported elements provides optional references to explicit representations of those element. But, it is not optional in reality. Explicit representation is required.

Recommendation: Correct as indicated.

3. Editorial comments and recommendations

(1) **Issue**: The term 'solid modeler' in page xii is not appropriate since use of current shape parametrics is not limited to solid model.

Recommendation: Change it to 'shape modeler'.

(2) **Issue**: The term 'hybrid models' is no where defined.

Recommendation: Check if use of 'hybrid of these two models' is appropriate.

(3) **Issue**: The term 'convert' is used in page-1. If it is meant to convert existing AP203 based model into parametric model, it is quite difficult since persistent entity name is nowhere considered in AP203 specification.

Recommendation: Change the whole sentence so that capabilities provided in this standard allows not only fully evaluated shape model representation but also the reason why the model is created as presented and related constraints.

(4) **Issue**: It is described in page-1 that mathematical expressions and functions are used in Clause 5. But in most advanced CAD systems, it is used in all three types of parametric representation; sketch, parametric assembly, and history based part shape modeling.

Recommendation: Change the sentence for reflecting practical use.

(5) **Issue**: Scope description in page-1 does not reflect the inclusion of 'Parametric Assembly' Example is 'Finally, ----'.

Recommendation: Change related description so that inclusion of parametric assembly can be clearly understood.

(6) **Issue**: The final out of scope item in page-2 is not appropriate.

Recommendation: Remove it or change the sentence so that the fact that neutral assembly model specification is not included but parametric assembly is included becomes clear.

(7) **Issue**: The term 'geometrically founded' is said to appear in 10303-42.

Recommendation: It should be changed to 10303-43 where the term is defined.

(8) **Issue**: Explanation of 3.5.25 instantiable (page10): is it appropriate English?
Recommendation: Let's check again.

(9) **Issue**: Similar terms generative model, procedural model, and history-based model appear in page 9 and 11.
Recommendation: From constraint point of view, explicit/implicit is a clear cut distinction. From procedure point of view, non-procedural/procedural is a clear cut distinction. Generative model or history-based model should be reconsidered if we can merge them into procedural model.

(10)**Issue**: The example of 3.5.36 may not be appropriate since it is very rare that a designer leave a manufacturing engineer the freedom of determining details of product shape. A designer should complete product shape satisfying all the design requirements. There are cases that a manufacturing engineer changes product shape passed from designers such as addition of fillet, or change of shape for taking into account spring back or over crown behavior. But, it is not a typical example of completing underconstrained shape.
Recommendation: A fundamental designer determines very basic product shape and constraints for satisfying key requirements. Details of the product shape is usually determined by detail designers.This could be a typical example of 3.5.36: underconstrained condition.

(11)**Issue**: It is written in the NOTE of 3.5.38 page-12 that the imposition of additional constraints to give a unique solution often requires the use of awkward and artificial methods, and the use of the current result to remove ambiguity is more practical for the purposes of ISO 10303.
But actually, parametrically modeled product shape such as a water jacket is quite complicated and imagination of unique solution from a current result is not realistic.
Recommendation: We should include some specification for enabling acquisition of a unique solution even if it is awkward or artificial.

(12)**Issue**: The term 'instantiable mathematics' is used in 4.4:Fundamental concepts and assumptions, page-15 which seems to be too much general term.
Recommendation: 'instantiable mathematical expression' seems better.

(13)**Issue**: The term 'Function entities defined ---' is used in 4.2.1 :Intelligent EXPRESS strings, page-15.
Recommendation: 'Functions defined ---' is recommended.

(14)**Issue**: The term 'maximum bound' is used in 4.2.2:Instance identifiers, page-16.
Recommendation: It should be changed to 'minimum bound'.

(15)**Issue**: The term 'An **express_in _string** entity is' is mistaken.
Recommendation: It should be changed to 'An **express_in_string** is'.

(16)**Issue**:4.3.11:**logical_expression** and 4.3.12: **boolean_expression** have same functionality except 'UNKNOWN' attribute.
Recommendation: It had better be merged into one expression, say, **logical_expression**.

(17)**Issue**: The necessity of 4.3.14: **entity_instance_expression** needs discussion.

Recommendation: I personally think it unnecessary since it is a case of express_function_call.

(18)**Issue**: Necessity of 4.3.15: **value_assignable_expression** and 4.3.16: **generic_expression** is questionable.

Recommendation: If 'D1 := #18.attributes(4)' style variable definition is allowed in 4.4.2: **variable_semantics** entity, then these may be unnecessary.

(19)**Issue**: The author prefers to use **model_variable** rather than **variable_semantics**.

Recommendation: Let's change the entity name to **model_variabl**e.

(20)**Issue**: It is described in 5.2.2:Current result that the current result may be used by the receiving system to determine the particular choices of constraint solutions, but as already described in (11), it is not sufficient to determine desired solution.

Recommendation: Same as that of (11).

(21)**Issue**: semantics attribute of 5.4.3 predefined_constraint is not given in its entity definition.

Recommendation: Add it if necessary.

(22)**Issue**: Duplicated word 'constraint constraint' in an example of 6.2.1

Recommendation: Change to 'constraint'.

(23)**Issue**: It is written in 6.2.2:Dimensionality of explicit geometric constraints(page-36) that foregoing distinction in interpretation of the constraints is not included in this resource schema; the appropriate specialization should be made in Application Protocols or Application Modules using it.

Recommendation: If this sentence implies the distinction of 2D and 3D explicit constraints, it should be changed since Parametric Assembly Constraint schema is included in Clause-7 of this standard.

(24)**Issue**: There is a miss typing as' unless the its dimensional subtype---' in 6.4.5: point_distance_constraint.

Recommendation: Correct it as' unless its dimensional subtype---'.

(25)**Issue**: The sentence ' If two smooth elements are involved the distance will be measured along their common normal' in 6.4.5 is difficult to understand.

Recommendation: More easy to understand explanation may be necessary.

(26)**Issue**: NOTE2 in 6.4.5 is questionable

It is explained that if only a single constrained point is specified, this constraint will convey no significant information unless the subtype **point_distance_constraint_with_dimension** is also instanced. However, such a case does not give rise to a logical inconsistency, and its inclusion by a formal proposition would complicate the definition of the subtype. But, it is the truth that single constrained point case is meaningless even if it does not give rise to logical inconsistency.

Same situations also appear in 6.4.7, 6.4.9, and 6.4.11.

Recommendation: The author would suggests to try formal specification . It should at least

be mentioned in informal proposition.

(27)**Issue**: EXPRESS usage in 6.4.22 projected_curve_constraint does not seem appropriate

Recommendation: 'SET[1:1] OF curve' is identical to just 'curve'.

'SET[1:2] OF geometry_element' should be 'SET[2] of geometry_element'.

(28)**Issue**: In WR2 of 6.4.33:implicit_curve, 'generation is by projection' is seen which should be modified to 'generation is projection'.

Recommendation: Correct as indicated.

(29): **Issue**: As described above, imported geometric element is not only curves but also points.

Recommendation: All the related descriptions had better be changed to imported elements(point or curve)

END of Comments